IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Davis et al.**	§	
	§	Group Art Unit: **2154**
Serial No. **09/981,873**	§	
	§	Examiner: **Patel, Ashokkumar B.**
Filed: **October 18, 2001**	§	
	§	
For: **Method and Apparatus for Partitioned Environment for Web Application Servers**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Reinstatement of Appeal, filed in this case on April 13, 2007.

No fees are believed to be required. If, however, any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-29.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: None.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1-29.
4. Claims allowed: None.
5. Claims rejected: 1-29.
6. Claims objected to: None.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-29.

STATUS OF AMENDMENTS

No amendments were submitted after the Final Office Action of March 7, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method in a data processing system for managing access to a set of applications associated with a universal resource locator (Specification, p. 8, lines 28-29). The method includes the steps of, receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification (Specification, p. 14, lines 11-14), modifying the universal resource locator based on the user identification (Specification, p. 15, lines 12-15), wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and directing the request to a selected application within the set of applications using the modified universal resource locator (Specification, p. 15, lines 7-15).

B. CLAIM 7 - INDEPENDENT

The subject matter of claim 7 is directed to a method in a data processing system for managing access to a plurality of applications (Specification, p. 8, lines 28-29). The method includes the steps of, associating the plurality of applications with a first universal resource locator (Specification, p. 14, lines 18-20), assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator (Specification, p. 14, lines 20-25), receiving a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user (Specification, p. 15, lines 15-18), modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and redirecting the request using the modified universal resource locator to a particular application within the plurality of applications (Specification, p. 15, lines 7-15).

C. CLAIM 10 - INDEPENDENT

The subject matter of claim 10 is directed to a data processing system (Specification, p. 1, lines 7-8). The data processing system comprises of a bus system (Specification, p. 10, line 9), a communications unit connected to the bus system (Specification, p. 10, lines 19-25) and a processing unit connected to the bus system (Specification, p. 10, lines 7-9). The processing unit executes the set of instructions to receive a request for a first application from a second application in which the request includes a universal resource locator and a user identification (Specification, p. 15, lines 15-18), modify the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and direct the request to a selected application within a set of applications using the modified universal resource locator(Specification, p. 15, lines 7-15).

D. CLAIM 11 - INDEPENDENT

The subject matter of claim 11 is directed to a data processing system (Specification, p. 1, lines 7-8). The data processing system comprises a bus system (Specification, p. 10, line 9); a communications unit connected to the bus system (Specification, p. 10, lines 19-25); a memory connected to the bus system (Specification, p. 10, lines 11-13), wherein the memory includes a set of instructions; and a processing unit connected to the bus system (Specification, p. 10, lines 7-9). The processing unit executes the set of instructions to associate a plurality of applications with a first universal resource locator (Specification, p. 14, lines 18-20), assign the plurality of applications with plurality of universal resource locators excluding the first universal resource locator (Specification, p. 14, lines 20-25), receive a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user (Specification, p. 15, lines 15-18), modify the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and redirect the request using the modified universal resource locator to a particular application within the plurality of applications (Specification, p. 15, lines 7-15).

E. CLAIM 12 - INDEPENDENT

The subject matter of claim 12 is directed to a data processing system for managing access to a set of applications associated with a universal resource locator (Specification, p. 8, lines 28-29). The data processing system comprises receiving means (Specification, p. 10, lines 7-9) for receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification (Specification, p. 15, lines 15-18), modifying means (Specification, p. 10, lines 7-9) for modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and directing means (Specification, p. 10, lines 7-9) for directing the request to a selected application within the set of applications using the modified universal resource locator (Specification, p. 15, lines 7-15).

F. CLAIM 15 - DEPENDENT

The subject matter of claim 15 is directed to the data processing system of claim 12. The data processing system further comprising: replacing means (Specification, p. 10, lines 7-9) for replacing the selected application with a new selected application (Specification, p. 9, lines 6-14).

G. CLAIM 17 - DEPENDENT

The subject matter of claim 17 is directed to the data processing system of claim 12. Each application within the set of applications is assigned to a different universal resource locator (Specification, p. 14, lines 8-14). The directing means (Specification, p. 10, lines 7-9) comprises first identifying means (Specification, p. 10, lines 7-9) for identifying the set of applications using the universal resource locator (Specification, p. 14, lines 4-7), second identifying means (Specification, p. 10, lines 7-9) for identifying the selected application based on the user identification (Specification, p. 14, lines 8-11), and sending means (Specification, p. 10, lines 7-9) for sending the request to the selected application using an assigned universal resource locator assigned to the selected applications (Specification, p. 14, lines 11-14).

H. CLAIM 18 - INDEPENDENT

The subject matter of claim 18 is directed to a data processing system for managing access to a plurality of applications (Specification, p. 8, lines 28-29). The data processing system comprises associating means (Specification, p. 10, lines 7-9) for, associating the plurality of applications with a first universal resource locator (Specification, p. 14, lines 18-20), assigning means (Specification, p. 10, lines 7-9) for assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator (Specification, p. 14, lines 20-25), receiving means (Specification, p. 10, lines 7-9) for receiving a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user (Specification, p. 15, lines 15-18), modifying means (Specification, p. 10, lines 7-9) for modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and redirecting means (Specification, p. 10, lines 7-9) for redirecting the request using the modified universal resource locator to a particular application within the plurality of applications (Specification, p. 15, lines 7-15).

I. CLAIM 21 - INDEPENDENT

The subject matter of claim 21 is directed to a computer program product in a computer readable medium (Specification, p. 19, lines 5-8) for managing access to a set of applications associated with a universal resource locator. The computer program product comprises of first instructions for receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification (Specification, p. 14, lines 11-14), second instructions for modifying the universal resource locator based on the user identification (Specification, p. 15, lines 12-15), wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and third instructions for directing the request to a selected application within the set of applications using the modified universal resource locator (Specification, p. 15, lines 7-15).

J. CLAIM 27 - INDEPENDENT

The subject matter of claim 27 is directed to a computer program product in a computer readable medium (Specification, p. 19, lines 5-8) for managing access to a plurality of applications (Specification, p. 8, lines 28-29). The computer program product comprises first instructions (Specification, p. 10, lines 7-9) for, associating the plurality of applications with a first universal resource locator (Specification, p. 14, lines 18-20), second instructions (Specification, p. 10, lines 7-9) for assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator (Specification, p. 14, lines 20-25), third instructions (Specification, p. 10, lines 7-9) for receiving a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user (Specification, p. 15, lines 15-18), fourth instructions (Specification, p. 10, lines 7-9) for modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application (Specification, p. 15, lines 9-11), and fifth instructions (Specification, p. 10, lines 7-9) for redirecting the request using the modified universal resource locator to a particular application within the plurality of applications (Specification, p. 15, lines 7-15).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 1-29)

Whether the rejection of claims 1-29 under 35 U.S.C. § 112, second paragraph, as being indefinite should be overturned.

B. GROUND OF REJECTION 2 (Claims 1-29)

Whether the rejection of claims 1-29 under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement should be overturned.

C. GROUND OF REJECTION 3 (Claims 1, 4-6, 10, 12, 15-17, 21, and 24-26)

Whether the rejection of claims 1, 4-6, 10, 12, 15-17, 21, and 24-26 under 35 U.S.C. § 102(b) as anticipated by *McDonough* et al., Controlling Access to Information, U.S. Patent No. 5,991,878 (November 23, 1999) (hereinafter "*McDonough*") should be overturned.

D. GROUND OF REJECTION 4 (Claims 2, 3, 13, 14, 22, and 23)

Whether the rejection of claims 2, 3, 13, 14, 22, and 23 under 35 U.S.C. § 103(a) as obvious over *McDonough* in view of *Levergood* et al., Internet Server Access Control and Monitoring Systems, U.S. Patent Application Publication 2006/0095526 (May 4, 2006) (hereinafter "*Levergood*") should be overturned.

E. GROUND OF REJECTION 5 (Claims 7, 11, 18, and 27)

Whether the rejection of claims 7, 11, 18, and 27 under 35 U.S.C. § 103(a) as obvious over *McDonough* in view of *Labarge* et al., Interface for Submitting Richly-Formatted Documents for Remote Processing, U.S. Patent Application Publication 2002/0188435 (December 12, 2002) (hereinafter "*Labarge*") should be overturned.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-29)

The Examiner rejects claims 1-29 under 35 U.S.C. § 112, second paragraph, as being indefinite. Appellants request that the Board of Patent Appeals and Interferences overturn this rejection and allow the claims. Claim 1 is representative of all claims in this grouping. Claim 1 is as follows:

> 1. A method in a data processing system for managing access to a set of applications associated with a universal resource locator, the method comprising:
>
> receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification;
>
> modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and
>
> directing the request to a selected application within the set of applications using the modified universal resource locator.

In regards to claim 1, the Examiner states:

> Referring to claim 1,
>
> Claim 1 recites:
>
> "a set of applications associated with a universal resource locator"
>
> "a request for a first application from a second application wherein the request includes the universal resource locator and a user identification"
>
> Claim thus states that the request from a second application received includes the universal resource locator and a user identification. This means that "a second application" is not included in the set of applications.
>
> Claim 1 further recites:
>
> "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; "
>
> How and why modifying has to occur on "the universal resource locator based on the user identification" when "the request from a second application is received" already "includes the universal resource locator and a user identification" and how" a selected application within the set of applications is related to the request when request is for "a first application from a second application and "a set of applications associated with a universal resource locator" is not including "a second application"?
>
> How is "the step of modifying maintains the universal resource locator unchanged as shown in the second application;" when the same claim limitation

recites "modifying the universal resource locator based on the user identification.", and when the request is for a first application from the second application includes the universal resource locator and a user identification" for "a first application"?

How is it determined to "maintain" the universal resource locator unchanged as shown in the second application"?

How is it determined "as shown in the second application" when "a set of applications associated with a universal resource locator" is not including "a second application" and when the request is for a first application from the second application includes the universal resource locator and a user identification" for "a first application"?

Now claim further recites:

"directing the request to a selected application within the set of applications using the modified the universal resource locator."

How the request is directed to "a selected application within the set of applications" when "a set of applications associated with a universal resource locator" and the request is for a first application? Thus, it is also unclear where "a selected application within the set of applications" is situated and the request is directed to finally "a selected application" when the request is for a first application.

And if the request has to be directed to a selected application within the set of applications using the modified universal resource locator, then what, if any, modifying the universal resource locator based on the user identification, would do when the request is for a first application?

Finally, how is it possible to direct a request using the modified the universal resource locator to a selected application within the set of applications wherein" a set of applications" is associated with "a universal resource locator" and the request is for a first application?

Final Office Action dated March 7, 2007, pp. 2-4.

Although Appellants believe that the Examiner's questions relate more to an enablement issue instead of an indefinite issue, Appellants will address the Examiner's questions one at a time.

The Examiner first ask "How and why modifying has to occur on "the universal resource locator based on the user identification" when "the request from a second application is received" already "includes the universal resource locator and a user identification." As to how modifying occurs, modification if performed by a proxy server. Proxy server 412 provides a mechanism for a partitioned development environment also referred to as a "sandbox" (Specification, p. 14, lines 5-7). Proxy server 412 assigns a copy of the application for each user and dynamically modifies

references to a particular instance or copy of the application based on which user is accessing this instance or copy (Specification, p. 14, lines 8-11).

As to why modifying has to occur when "the request from a second application is received" already "includes the universal resource locator and a user identification is because the included universal resource locator refers an application that several developers and analysts may simultaneously work on (Specification, p. 3, lines 21-23). Thus, each developer or analyst must be provided with their own copy/instance of the application for testing purposes (Specification, p. 9, lines 6-14). The copy/instance of the application assigned to the user is based on the user identification that is received with the universal resource locator (Specification, p. 14, lines 8-11).

The Examiner's questions following the preceding question are confusingly worded. Appellants believe the Examiner's line of questions fall into one of two categories of questions: 1) why is the request directed to a selected application within the set of applications using the modified the universal resource locator when the request is for a first application? And 2) how is it possible to direct a request using the modified the universal resource locator to a selected application within the set of applications when the "a set of applications" is associated with "a universal resource locator" and the request is for a first application?

In regards to question number 1, although the request is for a first application, the proxy server receives the requests from clients and provides the appropriate redirection of the requests to the copy of the application that is assigned to the user. The user requests a first application because programmers or developers access these applications using the same URL. Accessing the application using the same URL is important to development applications for the Web site because the development applications are simulated as if they are actually being served up or accessed on the Internet (Specification, p. 3, lines 23-25). However, because these applications may simultaneously be used by several developers, a copy/instance of the application must be provided to each developer for testing purposes (Specification, p. 9, lines 6-14). Thus, each developer is capable of simultaneous development on the same application using the same URL.

In regards to questions number 2, Appellants will give an example to clarify to the Examiner the features of claim 1 and show that claim 1 is not indefinite. Claim 1 recites that a URL is associated with a set of applications. For example, suppose the URL, "www.xyz.com", refers to a particular website. Now suppose copies of the xyz.com website are made, i.e., a set of

applications, and each copy is assigned to a particular developer. Thus these copies are associated, i.e., has some relation, to the URL. However, to get to these copies of the website, the user requests the original website, i.e., the first application, using the URL, "www.xyz.com", from the user's browser, i.e., the second application, and along with the URL requests the user's identification is received by the server. Based on the user's identification, the universal resource locator is modified to refer to a copy/instance of the application, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application, i.e., the user's browser Location field. Thus, the user is unaware that he/she is being directed to a copy/instance of the application. The proxy server directs the request to a selected application within the set of applications using the modified universal resource locator.

Appellants have shown that claim 1 is not indefinite and that based on the language of the claim in view of the Specification, Appellants have pointed out and distinctly claimed the subject matter which the Appellants regards as his invention. The Examiner has based his rejection to the remaining claims for similar reasons as stated in claim 1. Thus, the rejection of claim 1 and all the claims in this grouping are in error and the rejection under 35 U.S.C. § 112, second paragraph has been overcome.

B. GROUND OF REJECTION 2 (Claims 1-29)

The Examiner rejects claims 1-29 under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement. Appellants request that the Board of Patent Appeals and Interferences overturn this rejection and allow the claims. The Examiner states that:

> Claims 1, 7, 10 and their dependent claims are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.
>
> How is it possible "modifying" the URL can maintain the URL "unchanged"?

Final Office Action dated March 16, 2007, p. 10.

The test of enablement is whether one reasonably skilled in the art could make or use the invention from the disclosures in the patent coupled with information known in the art without undue experimentation. *United States v. Telectronics, Inc.*, 857 F.2d 778, 785, 8 USPQ2d 1217,

1223 (Fed. Cir. 1988), (See also MPEP § 2164.01). Claim 1, cited above, is representative of all the claims in this grouping.

The Examiner erroneously rejects claims 1-29 based on the Examiner's belief that is not possible to "modifying" the URL and maintain the URL unchanged. However, claim 1 recites "wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application." Thus, Appellants are not claiming that the URL is modified and unchanged because that would not be possible, as stated by the Examiner. Instead, claim 1 recites that the step of modifying the URL maintains the URL unchanged as shown in the second application. This feature of claim 1 is clearly possible to one of reasonable skill in the art.

The attached documentation entitled "Apache 1.3 URL Rewriting Guide"[1] describes how one can use Apache's mod_rewrite to solve typical URL-based problems with which webmasters are often confronted. The Apache module mod_rewrite is a sophisticated module which provides a powerful way to perform URL manipulations. The relevant portion of the document is as follows:

> **Content Handling**
>
> ***From Old to New (intern)***
>
> **Description:**
> Assume we have recently renamed the page foo.html to bar.html and now want to provide the old URL for backward compatibility. Actually we want that users of the old URL even not recognize that the pages was renamed.
> **Solution:**
> We rewrite the old URL to the new one internally via the following rule:
>
> RewriteEngine on
> RewriteBase /~quux/
> RewriteRule ^**foo**\.html$ **bar**.html
>
> ***From Old to New (extern)***
>
> **Description:**
> Assume again that we have recently renamed the page foo.html to bar.html and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.
> **Solution:**
> We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

[1] Ralf S. Engelschall, *Apache 1.3 URL Rewriting Guide*, December 1997, <rse@apache.org>

RewriteEngine on
RewriteBase /~quux/
RewriteRule ^**foo**\.html$ **bar**.html **[R]**

Engelschall, Apache 1.3 URL Rewriting Guide, pp. 11-12.

The first portion above "From Old to New (intern)" describes a possible method to implement the recited feature of claim 1. The user inputs in the old URL <foo.html>, the web server modifies /rewrites that URL internally to the new URL < bar.html>. The user does not recognize the switch because the URL is unchanged in the user's browser Location field, i.e., the URL is unchanged as shown in the second application.

The second part "From Old to New (extern)" states that the users of the old URL should be hinted to the new URL. Therefore, the user's browser Location field should change, too.

Thus, the first portion above clearly demonstrates that one can modify the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application as recited in claim 1. Therefore, the Examiner's rejection of claim 1-29 under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement is clearly erroneous and should be overturned.

C. GROUND OF REJECTION 3 (Claims 1, 4-6, 10, 12, 15-17, 21, and 24-26)

The Examiner rejects claims 1, 4-6, 10, 12, 15-17, 21, and 24-26 under 35 U.S.C. § 102(b) as anticipated by *McDonough*. Appellants request that the Board of Patent Appeals and Interferences overturn this rejection and allow the claims.

C.1. Claims 1, 4-6, 10, 12, 15-17, 21, and 24-26

C.1.i. Response to Rejection

Claim 1 is representative of this grouping of claims. Regarding claim 1, the Examiner asserts the following:

> Referring to claim 1,
> McDonough teaches a method in a data processing system for managing access to a set of applications associated with a universal resource locator (Fig. 1, col. 2, line 48-52), the method comprising:

receiving a request, wherein the request includes the universal resource locator (col. 2, line 58-61, col. 3, line 15-20) and a user identification (col. 3, line 39-44, col. 4, line 64-col. 5, line 6) ; and

directing the request to a selected application within the set of applications using the universal resource locator and the user identification. (col.5, line 7-30).

Final Office Action dated March 7, 2007, p. 14.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims. Claim 1 is as previously presented is as follows:

1. A method in a data processing system for managing access to a set of applications associated with a universal resource locator, the method comprising:

receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification;

modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and

directing the request to a selected application within the set of applications using the modified universal resource locator.

McDonough does not anticipate claim 1 as previously presented because *McDonough* does not teach "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator *unchanged as shown in the second application*" and "directing the request to a selected application within the set of applications using the *modified* universal resource locator." Instead, *McDonough* teaches a method for controlling access to information in a distributed computing system. The method includes receiving a request for information accompanied by encrypted session state data, such as when provided as a generic

cookie. The method also includes, based on the encrypted session state data, determining whether to pass the request to a source of the information. The source can be a server computer. (*McDonough*, col. 1, lines 34-41).

Furthermore, *McDonough* receives a URL-based request for information from the browser software. The request includes a data packet made up of the URL and other computer data. (*McDonough,* col. 3, lines 15-18). Based on the request, a set of application software instances is identified. (*McDonough*, col. 5, lines 7-8). Based on the authorized application codes and the authorized realm indicator, derived from the smart cookie, the process determines whether the user is authorized to have access to the identified set. For example, the authorized realm indicator may include "/test" and the authorized applications codes may include "GH" and "IJ". In such a case, if the URL includes "/working/GH", the authorized realm indicator blocks access to a corresponding "GH" application software instance even though the authorized applications codes include "GH". If the user is not authorized to have access, the browser software is provided with a response indicating that access is denied. (*McDonough*, col. 5, lines 15-26).

As shown in the above portion of *McDonough*, *McDonough* does not teach modifying the universal resource locator based on the user identification. *McDonough* receives a universal resource locator and a cookie. Based off the cookie, *McDonough* determines whether the user is authorized to have access to the identified set associated with the universal resource locator. *McDonough* does not teach "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application."

Furthermore, because *McDonough* does not modify the universal resource locator based on the user identification, *McDonough* inherently cannot direct the request to a selected application within the set of applications using the *modified* universal resource locator. Consequently, *McDonough* does not teach this claimed feature. Therefore, *McDonough* does not anticipate claim 1 or any other claim in this grouping of claims.

C.1.ii. Rebuttal of Examiner's Response

The Examiner provides no response to the above argument other than asserting that the amended features of claim 1 fails under 35 U.S.C. § 112. (Final Office Action dated March 7, 2007, p. 2).

D. GROUND OF REJECTION 4 (Claims 2, 3, 13, 14, 22, and 23)

The Examiner rejects claims 2, 3, 13, 14, 22, and 23 under 35 U.S.C. § 103(a) as obvious over *McDonough* in view of *Levergood.* Appellants request that the Board of Patent Appeals and Interferences overturn this rejection and allow the claims.

D.1. Claims 2, 3, 13, 14, 22, and 23

D.1.i. Response to Rejection

Claim 2 is representative of this grouping of claims. Regarding claim 2, the Examiner asserts the following:

> Referring to claims 2 and 3,
>
> Keeping in mind the teachings of McDonough, McDonough specifically fails to teach the method of claim 1 , wherein the user identification is an internet Protocol address of a node originating the request, and the method of claim 1, wherein the user identification is a user name located within the request.
>
> Levergood teaches at para., [0031] If the initial GET URL contains a SID, the content server determines whether the request is directed to a page within the current domain 106.", para.. [0014] In the preferred embodiment, a valid SID allows the client to access all controlled files within a protection domain without requiring further authorization. A protection domain is defined by the service provider and is a collection of controlled files of common protection within one or more servers., para. [0012], "A valid SID typically comprises a user identifier, an accessible domain, a key identifier, an expiration time such as date, the IP address of the user computer, and an unforgeable digital signature such as a cryptographic hash of all of the other items in the SID encrypted with a secret key. "
>
> The SID (a session identification) including these many user data including user name and user's IP address as well as appending the SID to the initial request to control the access to the information would be so recognized by persons of ordinary skill, such that it would have been obvious for one in ordinary skill in the art at the time the invention was made to add the teachings of Levergood's session identification data such as user name and IP address into the McDonough's encrypted session state data provided as a "PRIVATE cookie.

> It would have been obvious because the SID allows the user to access specific document with the identification of knowing exactly who the client is and what it's IP address is as taught by Levergood.

Office action dated September 07, 2006 pages 6-7.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The scope and content of the prior art are... determined; differences between the prior art and the claims at issue are... ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue. *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *Id.* (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006)).

In the case at hand, the cited references when considered as a whole do not teach or suggest all of the limitations of the claims, arranged as they are in the claims. Claim 2 is as follows:

> 2. The method of claim 1, wherein the user identification is an Internet Protocol address of a node originating the request.

The Examiner cannot establish a *prima facie* obviousness rejection against claim 2 using the present references because neither *McDonough* nor *Levergood* teach or suggest all features of claim 1, from which claim 2 depends. As discussed above, *McDonough* does not teach the claimed feature, "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application." Because *McDonough* does not teach any type of modification to the universal resource locator entered by the user, *McDonough* also does not suggest the features of claim 1.

Furthermore, *Levergood* also does not teach or suggest all of the features of claim 1. *Levergood* relates to methods for controlling and monitoring access to network servers. *Levergood*

teaches forwarding a service request (URL) from the client to the server and appending a session identification (SID) to the request and to subsequent service requests from the client to the server within a session of requests (*Levergood*, paragraph 0011, lines 6-10). However, *Levergood* does not teach "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application" as recited in claim 1. *Levergood* specifically states the *browser on the client computer* executes a relative link by rewriting the current URL to *replace* the old controlled page name with a new one (*Levergood*, page 2, paragraph 0015, lines 7-11). Because the clients browser rewrites and replaces the old universal resource locator with the new one, *Levergood* also does not suggest the feature, "modifying the universal resource locator based on the user identification, *wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application*" as in claim 1.

Because neither *McDonough* nor *Levergood* teach or suggest all of the features of claim 1, and because claim 2 depends from claim 1, the proposed combination of *McDonough* and *Levergood* when considered as a whole does not teach or suggest all of the features of claim 2. Accordingly, the Examiner cannot establish a *prima facie* obviousness rejection of claim 2 or any other claim in this grouping of claims based on the cited references.

D.1.ii. Rebuttal of Examiner's Response

The Examiner provides no response to the above argument other than asserting that the amended features of claim 1 fails under 35 U.S.C. § 112. (Final Office Action dated March 7, 2007, p. 2).

E. GROUND OF REJECTION 5 (Claims 7, 11, 18, and 27)

The Examiner rejects claims 7, 11, 18, and 27 under 35 U.S.C. § 103(a) as obvious over *McDonough* in view of *Labarge*. Appellants request that the Board of Patent Appeals and Interferences overturn this rejection and allow the claims.

E.1. Claims 7, 11, 18, and 27

E.1.i. Response to Rejection

Claim 7 is representative of this grouping of claims. Regarding claim 7, the Examiner asserts the following:

> Referring to claim 7,
>
> McDonough teaches a method in a data processing system for managing access to a plurality of applications (co1.2, line 48-52, Fig.1), the method comprising:
>
> associating the plurality of applications with a first universal resource locator (col. 2, line 53-57);
>
> receiving a request including the first universal resource locator (col. 2, line 58- 61, col. 3, line 15-20) and an identification of a user (col. 3, line 39-44, col. 4, line 64- co1.5, line 6); and
>
> redirecting the request using the first universal resource locator to a particular application within the plurality of applications using a particular universal resource locator associated with ,the particular application based on the identification (co1.5, line 7-30).
>
> McDonough fails to teach assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator.
>
> Labarge teaches at para. [0039, " Preferably, redirection provides flexibility for the URL addresses assigned to various machine translation servers. Only the redirection server URL needs to remain constant for access by the word processing application 21 0, while the machine translation server URLs may be changed from time to time to reflect configuration changes or to reflect changes in available machine translation services. For example, if the ABC Translation Company goes out of business, the URL address for the ABC Translation Company will be removed and/or replaced by the URL of an alternate machine translation service designated for a given translation service, for example, "Japanese to English."
>
> Labarge's teachings would be so recognized by persons of ordinary skill, such that it would have been obvious for one in ordinary skill in the art at the time the invention was made to implement these teachings at the McDonough's web server computer such that the flexibility is achieved in assigning the URL addressees to various application instances with keeping the web server's address (URL) constant and then redirecting the McDonough's application user to the appropriate application instance based on the determination of access authorization.
>
> This would have been obvious because it provides the flexibility wherein application URLs may be changed from time to time, with no user's are affected, and server UL can be kept constant.

Final Office Action dated March 7, 2007, p. 19-20.

The Examiner cannot establish a *prima facie* obviousness rejection based on the cited references because the proposed combination does not teach or suggest all of the features of claim 7. In the case at hand, the teachings of the references themselves do not teach or suggest the claimed subject matter to a person of ordinary skill in the art. Claim 7 as previously presented is as follows:

> 7. A method in a data processing system for managing access to a plurality of applications, the method comprising:
> associating the plurality of applications with a first universal resource locator;
> assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator;
> receiving a request for a first application from a second application, wherein the request includes including the first universal resource locator and an identification of a user; and
> modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application; and
> redirecting the request using the first modified universal resource locator to a particular application within the plurality of applications.

The Examiner cannot establish a *prima facie* obviousness rejection based on the cited references because neither *McDonough* nor *Labarge* teach or suggest all features of claim 7 as amended. As discussed above, *McDonough* does not teach or suggest the claimed feature, "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application."

Additionally, *Labarge* also does not teach or suggest this claimed feature. *Labarge* provides an application programming interface (API) for submitting a richly-formatted text selection or document to a remote machine translation server for translation. The user's software application program, such as a word processor, requests from a redirector server the uniform resource locator (URL) of a remote machine translation server. All formatting, images, including pictures, drawings and the like, and other data objects not requiring translation are saved in a temporary file on the user's computer. The user's word processing software application opens an instance of the user's Internet browser, and the Internet browser submits the text selection to the

remote machine translation server via the Internet, an intranet, or other distributed computing environment (*Labarge,* paragraph 0007).

More particularly, once the user selects a translation service, for example, "Japanese to English", a redirection request is sent from the word processing application to a redirection server. The purpose for the redirection request is to obtain the URL for the particular translation service selected by the user, for example, "Japanese to English" (*Labarge*, paragraph 0038).

Thus, *Labarge* does not teach "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application" as in claim 7 as amended. Instead, *Labarge* teaches a software application program requesting a URL from a redirector server based on the translation service selected by the user. *Labarge* does not modify a universal resource locator *based on the user identification* because *Labarge* selects a URL based on the translation service selected by the user and not by the user identification. Because *Labarge* does not teach the above feature, *Labarge* also does not suggest the feature, "modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application" as in claim 7 as amended.

Because neither *McDonough* nor *Labarge* teach or suggest all of the features of claim 7 as amended, the proposed combination of *McDonough* and *Labarge* when considered as a whole does not teach or suggest all of the features of claim 7 as amended. Accordingly, the Examiner cannot establish a *prima facie* obviousness rejection of claim 7 or any other claim in this grouping of claims based on the cited references.

E.1.ii. Rebuttal of Examiner's Response

The Examiner provides no response to the above argument other than asserting that the amended features of claim 1 fails under 35 U.S.C. § 112. (Final Office Action dated March 7, 2007, p. 2).

F. CONCLUSION

Appellants have shown that the Examiner's rejections of claims 1-29 under 35 U.S.C. § 112 is in error and should be overturned. Additionally, the Examiner failed to state a *prima facie* obviousness rejection in view of the combinations of *McDonough*, *Levergood*, and *Labarge* because the proposed combinations do not teach or suggest all of the features of the claims, and because of the other reasons presented above.

Therefore, Appellants respectfully request that the Board of Patent Appeals and Interferences overturn the rejections. Additionally, Appellants request the Board to allow the claims.

/Theodore D. Fay III/
Theodore D. Fay III
Reg. No. 48,504
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

TDF/nh

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method in a data processing system for managing access to a set of applications associated with a universal resource locator, the method comprising:

receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification;

modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and

directing the request to a selected application within the set of applications using the modified universal resource locator.

2. The method of claim 1, wherein the user identification is an Internet Protocol address of a node originating the request.

3. The method of claim 1, wherein the user identification is a user name located within the request.

4. The method of claim 1 further comprising: replacing the selected application with a new selected application.

5. The method of claim 4, wherein the new selected application is a new version of the selected application.

6. The method of claim 1, wherein each application within the set of applications is assigned to a different universal resource locator and wherein the directing step comprises:

identifying the set of applications using the universal resource locator;

identifying the selected application based on the user identification; and

sending the request to the selected application using an assigned universal resource locator assigned to the selected applications.

7. A method in a data processing system for managing access to a plurality of applications, the method comprising:

associating the plurality of applications with a first universal resource locator;

assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator;

receiving a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user;

modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application; and

redirecting the request using the modified universal resource locator to a particular application within the plurality of applications.

8. The method of claim 7, wherein the identification is an Internet Protocol address.

9. The method of claim 7, wherein the identification is a user name.

10. A data processing system comprising:

a bus system;

a communications unit connected to the bus system; a memory connected to the bus system, wherein the memory includes a set of instructions; and

a processing unit connected to the bus system, wherein the processing unit executes the set of instructions to receive a request for a first application from a second application in which the request includes a universal resource locator and a user identification; modify the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and direct the request to a selected application within a set of applications using the modified universal resource locator.

11. A data processing system comprising:

a bus system;

a communications unit connected to the bus system;

a memory connected to the bus system, wherein the memory includes a set of instructions; and

a processing unit connected to the bus system, wherein the processing unit executes the set of instructions to associate a plurality of applications with a first universal resource locator; assign the plurality of applications with plurality of universal resource locators excluding the first

universal resource locator; receive a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user; modify the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application; and redirect the request using the modified universal resource locator to a particular application within the plurality of applications.

12. A data processing system for managing access to a set of applications associated with a universal resource locator, the data processing system comprising:

receiving means for receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification;

modifying means for modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and

directing means for directing the request to a selected application within the set of applications using the modified universal resource locator.

13. The data processing system of claim 12, wherein the user identification is an Internet Protocol address of a node originating the request.

14. The data processing system of claim 12, wherein the user identification is a user name located within the request.

15. The data processing system of claim 12 further comprising: replacing means for replacing the selected application with a new selected application.

16. The data processing system of claim 15, wherein the new selected application is a new version of the selected application.

17. The data processing system of claim 12, wherein each application within the set of applications is assigned to a different universal resource locator and wherein the directing means comprises:

first identifying means for identifying the set of applications using the universal resource locator;

second identifying means for identifying the selected application based on the user identification; and

sending means for sending the request to the selected application using an assigned universal resource locator assigned to the selected applications.

18. A data processing system for managing access to a plurality of applications, the data processing system comprising:

associating means for associating the plurality of applications with a first universal resource locator;

assigning means for assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator;

receiving means for receiving a request for a first application from a second application,

wherein the request includes the first universal resource locator and an identification of a user;

modifying means for modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application; and

redirecting means for redirecting the request using the modified universal resource locator to a particular application within the plurality of applications.

19. The data processing system of claim 18, wherein the identification is an Internet Protocol address.

20. The data processing system of claim 18, wherein the identification is a user name.

21. A computer program product in a computer readable medium for managing access to a set of applications associated with a universal resource locator, the computer program product comprising:

first instructions for receiving a request for a first application from a second application, wherein the request includes the universal resource locator and a user identification;

second instructions for modifying the universal resource locator based on the user identification, wherein the step of modifying maintains the universal resource locator unchanged as shown in the second application; and

third instructions for directing the request to a selected application within the set of applications using the modified universal resource locator and the user identification.

22. The computer program product of claim 21, wherein the user identification is an Internet Protocol address of a node originating the request.

23. The computer program product of claim 21, wherein the user identification is a user name located within the request.

24. The computer program product of claim 21 further comprising:

third instructions for replacing the selected application with a new selected application.

25. The computer program product of claim 24, wherein the new selected application is a new version of the selected application.

26. The computer program product of claim 21, wherein each application within the set of applications is assigned to a different universal resource locator and wherein the second instructions comprises:

first sub-instructions for identifying the set of applications using the universal resource locator;

second sub-instructions for identifying the selected application based on the user identification; and

third sub-instructions for sending the request to the selected application using an assigned universal resource locator assigned to the selected applications.

27. A computer program product in a computer readable medium for managing access to a plurality of applications, the computer program product comprising:

first instructions for associating the plurality of applications with a first universal resource locator;

second instructions for assigning the plurality of applications with plurality of universal resource locators excluding the first universal resource locator;

third instructions for receiving a request for a first application from a second application, wherein the request includes the first universal resource locator and an identification of a user;

fourth instructions for modifying means for modifying the first universal resource locator based on the user identification, wherein the step of modifying maintains the first universal resource locator unchanged as shown in the second application; and

fifth instructions for redirecting the request using the modified universal resource locator to a particular application within the plurality of applications.

28. The computer program product of claim 27, wherein the identification is an Internet Protocol address.

29. The computer program product of claim 27, wherein the identification is a user name.

EVIDENCE APPENDIX

Apache 1.3, URL Writing Guide, a copy of which is attached hereto, is available at http://httpd.apache.org/docs/1.3/misc/rewriteguide.html.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

Apache HTTP Server Version 1.3

Is this the version you want? For more recent versions, check our documentation index.

Apache 1.3 URL Rewriting Guide

Originally written by
Ralf S. Engelschall <rse@apache.org>
December 1997

This document supplements the mod_rewrite reference documentation. It describes how one can use Apache's mod_rewrite to solve typical URL-based problems with which webmasters are often confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.

Introduction to mod_rewrite

The Apache module mod_rewrite is a killer one, i.e. it is a really sophisticated module which provides a powerful way to do URL manipulations. With it you can do nearly all types of URL manipulations you ever dreamed about. The price you have to pay is to accept complexity, because mod_rewrite's major drawback is that it is not easy to understand and use for the beginner. And even Apache experts sometimes discover new aspects where mod_rewrite can help.

In other words: With mod_rewrite you either shoot yourself in the foot the first time and never use it again or love it for the rest of your life because of its power. This paper tries to give you a few initial success events to avoid the first case by presenting already invented solutions to you.

Practical Solutions

Here come a lot of practical solutions I've either invented myself or collected from other peoples solutions in the past. Feel free to learn the black magic of URL rewriting from these examples.

ATTENTION: Depending on your server-configuration it can be necessary to slightly change the examples for your situation, e.g. adding the [PT] flag when additionally using mod_alias and mod_userdir, etc. Or rewriting a ruleset to fit in `.htaccess` context instead of per-server context. Always try to understand what a particular ruleset really does before you use it in order to avoid problems.

URL Layout

Canonical URLs

Description:

On some webservers there are more than one URL for a resource. Usually there are canonical URLs (which should be actually used and distributed) and those which are just shortcuts, internal ones, etc. Independent which URL the user supplied with the request he should finally see the canonical one only.

Solution:

We do an external HTTP redirect for all non-canonical URLs to fix them in the location view of the Browser and for all subsequent requests. In the example ruleset below we replace `/~user` by the canonical `/u/user` and fix a missing trailing slash for `/u/user`.

```
RewriteRule   ^/~([^/]+)/?(.*)     /u/$1/$2   [R]
RewriteRule   ^/([uge])/([^/]+)$   /$1/$2/    [R]
```

Canonical Hostnames

Description:

The goal of this rule is to force the use of a particular hostname, in preference to other hostnames which may be used to reach the same site. For example, if you wish to force the use of **www.example.com** instead of **example.com**, you might use a variant of the following recipe.

Solution:

```
# For sites running on a port other than 80
RewriteCond %{HTTP_HOST}   !^fully\.qualified\.domain\.name [NC]
RewriteCond %{HTTP_HOST}   !^$
RewriteCond %{SERVER_PORT} !^80$
RewriteRule ^/(.*)         http://fully.qualified.domain.name:%{SERVER_PO

# And for a site running on port 80
RewriteCond %{HTTP_HOST}   !^fully\.qualified\.domain\.name [NC]
RewriteCond %{HTTP_HOST}   !^$
RewriteRule ^/(.*)         http://fully.qualified.domain.name/$1 [L,R]
```

Moved DocumentRoot

Description:

Usually the DocumentRoot of the webserver directly relates to the URL ``/''. But often this data is not really of top-level priority, it is perhaps just one entity of a lot of data pools. For instance at our Intranet sites there are `/e/www/` (the homepage for WWW), `/e/sww/` (the homepage for the Intranet) etc. Now because the data of the DocumentRoot stays at `/e/www/` we had to make sure that all inlined images and other stuff inside this data pool work for subsequent requests.

Solution:

We just redirect the URL `/` to `/e/www/`. While is seems trivial it is actually trivial with mod_rewrite, only. Because the typical old mechanisms of URL *Aliases* (as

provides by mod_alias and friends) only used *prefix* matching. With this you cannot do such a redirection because the DocumentRoot is a prefix of all URLs. With mod_rewrite it is really trivial:

```
RewriteEngine on
RewriteRule   ^/$  /e/www/  [R]
```

Trailing Slash Problem

Description:

Every webmaster can sing a song about the problem of the trailing slash on URLs referencing directories. If they are missing, the server dumps an error, because if you say `/~quux/foo` instead of `/~quux/foo/` then the server searches for a *file* named `foo`. And because this file is a directory it complains. Actually is tries to fix it themself in most of the cases, but sometimes this mechanism need to be emulated by you. For instance after you have done a lot of complicated URL rewritings to CGI scripts etc.

Solution:

The solution to this subtle problem is to let the server add the trailing slash automatically. To do this correctly we have to use an external redirect, so the browser correctly requests subsequent images etc. If we only did a internal rewrite, this would only work for the directory page, but would go wrong when any images are included into this page with relative URLs, because the browser would request an in-lined object. For instance, a request for `image.gif` in `/~quux/foo/index.html` would become `/~quux/image.gif` without the external redirect!

So, to do this trick we write:

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo$  foo/  [R]
```

The crazy and lazy can even do the following in the top-level `.htaccess` file of their homedir. But notice that this creates some processing overhead.

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteCond    %{REQUEST_FILENAME}  -d
RewriteRule    ^(.+[^/])$           $1/  [R]
```

Webcluster through Homogeneous URL Layout

Description:

We want to create a homogenous and consistent URL layout over all WWW servers on a Intranet webcluster, i.e. all URLs (per definition server local and thus server dependent!) become actually server *independed*! What we want is to give the WWW namespace a consistent server-independend layout: no URL should have to include any physically correct target server. The cluster itself should drive us automatically to the physical target host.

Solution:

First, the knowledge of the target servers come from (distributed) external maps which contain information where our users, groups and entities stay. The have the form

```
user1  server_of_user1
user2  server_of_user2
:      :
```

We put them into files `map.xxx-to-host`. Second we need to instruct all servers to redirect URLs of the forms

```
/u/user/anypath
/g/group/anypath
/e/entity/anypath
```

to

```
http://physical-host/u/user/anypath
http://physical-host/g/group/anypath
http://physical-host/e/entity/anypath
```

when the URL is not locally valid to a server. The following ruleset does this for us by the help of the map files (assuming that server0 is a default server which will be used if a user has no entry in the map):

```
RewriteEngine on

RewriteMap      user-to-host   txt:/path/to/map.user-to-host
RewriteMap     group-to-host   txt:/path/to/map.group-to-host
RewriteMap    entity-to-host   txt:/path/to/map.entity-to-host

RewriteRule   ^/u/([^/]+)/?(.*)   http://${user-to-host:$1|server0}/u/$1/$
RewriteRule   ^/g/([^/]+)/?(.*)  http://${group-to-host:$1|server0}/g/$1/$
RewriteRule   ^/e/([^/]+)/?(.*) http://${entity-to-host:$1|server0}/e/$1/$

RewriteRule   ^/([uge])/([^/]+)/?$          /$1/$2/.www/
RewriteRule   ^/([uge])/([^/]+)/([^.]+.+)   /$1/$2/.www/$3\
```

Move Homedirs to Different Webserver

Description:

A lot of webmaster aksed for a solution to the following situation: They wanted to redirect just all homedirs on a webserver to another webserver. They usually need such things when establishing a newer webserver which will replace the old one over time.

Solution:

The solution is trivial with mod_rewrite. On the old webserver we just redirect all `/~user/anypath` URLs to `http://newserver/~user/anypath`.

```
RewriteEngine on
RewriteRule   ^/~(.+)  http://newserver/~$1  [R,L]
```

Structured Homedirs

Description:

Some sites with thousend of users usually use a structured homedir layout, i.e. each homedir is in a subdirectory which begins for instance with the first character of the username. So, `/~foo/anypath` is `/home/f/foo/.www/anypath` while `/~bar/anypath` is `/home/b/bar/.www/anypath`.

Solution:

We use the following ruleset to expand the tilde URLs into exactly the above layout.

```
RewriteEngine on
RewriteRule   ^/~(([a-z])[a-z0-9]+)(.*)  /home/$2/$1/.www$3
```

Filesystem Reorganisation

Description:

This really is a hardcore example: a killer application which heavily uses per-directory `RewriteRules` to get a smooth look and feel on the Web while its data structure is never touched or adjusted. Background: ***net.sw*** is my archive of freely available Unix software packages, which I started to collect in 1992. It is both my hobby and job to to this, because while I'm studying computer science I have also worked for many years as a system and network administrator in my spare time. Every week I need some sort of software so I created a deep hierarchy of directories where I stored the packages:

```
drwxrwxr-x   2 netsw  users    512 Aug  3 18:39 Audio/
drwxrwxr-x   2 netsw  users    512 Jul  9 14:37 Benchmark/
drwxrwxr-x  12 netsw  users    512 Jul  9 00:34 Crypto/
drwxrwxr-x   5 netsw  users    512 Jul  9 00:41 Database/
drwxrwxr-x   4 netsw  users    512 Jul 30 19:25 Dicts/
drwxrwxr-x  10 netsw  users    512 Jul  9 01:54 Graphic/
drwxrwxr-x   5 netsw  users    512 Jul  9 01:58 Hackers/
drwxrwxr-x   8 netsw  users    512 Jul  9 03:19 InfoSys/
drwxrwxr-x   3 netsw  users    512 Jul  9 03:21 Math/
drwxrwxr-x   3 netsw  users    512 Jul  9 03:24 Misc/
drwxrwxr-x   9 netsw  users    512 Aug  1 16:33 Network/
drwxrwxr-x   2 netsw  users    512 Jul  9 05:53 Office/
drwxrwxr-x   7 netsw  users    512 Jul  9 09:24 SoftEng/
drwxrwxr-x   7 netsw  users    512 Jul  9 12:17 System/
drwxrwxr-x  12 netsw  users    512 Aug  3 20:15 Typesetting/
drwxrwxr-x  10 netsw  users    512 Jul  9 14:08 X11/
```

In July 1996 I decided to make this archive public to the world via a nice Web interface. "Nice" means that I wanted to offer an interface where you can browse directly through the archive hierarchy. And "nice" means that I didn't wanted to change anything inside this hierarchy - not even by putting some CGI scripts at the top of it. Why? Because the above structure should be later accessible via FTP as well, and I didn't want any Web or CGI stuff to be there.

Solution:

The solution has two parts: The first is a set of CGI scripts which create all the pages at all directory levels on-the-fly. I put them under `/e/netsw/.www/` as follows:

```
-rw-r--r--   1 netsw  users    1318 Aug  1 18:10 .wwwacl
drwxr-xr-x  18 netsw  users     512 Aug  5 15:51 DATA/
-rw-rw-rw-   1 netsw  users  372982 Aug  5 16:35 LOGFILE
-rw-r--r--   1 netsw  users     659 Aug  4 09:27 TODO
-rw-r--r--   1 netsw  users    5697 Aug  1 18:01 netsw-about.html
-rwxr-xr-x   1 netsw  users     579 Aug  2 10:33 netsw-access.pl
-rwxr-xr-x   1 netsw  users    1532 Aug  1 17:35 netsw-changes.cgi
-rwxr-xr-x   1 netsw  users    2866 Aug  5 14:49 netsw-home.cgi
drwxr-xr-x   2 netsw  users     512 Jul  8 23:47 netsw-img/
-rwxr-xr-x   1 netsw  users   24050 Aug  5 15:49 netsw-lsdir.cgi
-rwxr-xr-x   1 netsw  users    1589 Aug  3 18:43 netsw-search.cgi
-rwxr-xr-x   1 netsw  users    1885 Aug  1 17:41 netsw-tree.cgi
-rw-r--r--   1 netsw  users     234 Jul 30 16:35 netsw-unlimit.lst
```

The `DATA/` subdirectory holds the above directory structure, i.e. the real ***net.sw*** stuff and gets automatically updated via `rdist` from time to time. The second part of the problem remains: how to link these two structures together into one smooth-looking URL tree? We want to hide the `DATA/` directory from the user while running the appropriate CGI scripts for the various URLs. Here is the solution: first I put the following into the per-directory configuration file in the Document Root of the server to rewrite the announced URL `/net.sw/` to the internal path `/e/netsw`:

```
RewriteRule  ^net.sw$       net.sw/        [R]
RewriteRule  ^net.sw/(.*)$  e/netsw/$1
```

The first rule is for requests which miss the trailing slash! The second rule does the real thing. And then comes the killer configuration which stays in the per-directory config file `/e/netsw/.www/.wwwacl`:

```
Options       ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

#  we are reached via /net.sw/ prefix
RewriteBase   /net.sw/

#  first we rewrite the root dir to
#  the handling cgi script
RewriteRule   ^$                      netsw-home.cgi      [L]
RewriteRule   ^index\.html$           netsw-home.cgi      [L]

#  strip out the subdirs when
#  the browser requests us from perdir pages
RewriteRule   ^.+/(netsw-[^/]+/.+)$   $1                  [L]

#  and now break the rewriting for local files
RewriteRule   ^netsw-home\.cgi.*      -                   [L]
RewriteRule   ^netsw-changes\.cgi.*   -                   [L]
RewriteRule   ^netsw-search\.cgi.*    -                   [L]
RewriteRule   ^netsw-tree\.cgi$       -                   [L]
RewriteRule   ^netsw-about\.html$     -                   [L]
RewriteRule   ^netsw-img/.*$          -                   [L]

#  anything else is a subdir which gets handled
#  by another cgi script
RewriteRule   !^netsw-lsdir\.cgi.*    -                   [C]
```

```
RewriteRule    (.*)                          netsw-lsdir.cgi/$1
```

Some hints for interpretation:

1. Notice the L (last) flag and no substitution field ('-') in the forth part
2. Notice the ! (not) character and the C (chain) flag at the first rule in the last part
3. Notice the catch-all pattern in the last rule

NCSA imagemap to Apache mod_imap

Description:

When switching from the NCSA webserver to the more modern Apache webserver a lot of people want a smooth transition. So they want pages which use their old NCSA `imagemap` program to work under Apache with the modern `mod_imap`. The problem is that there are a lot of hyperlinks around which reference the `imagemap` program via `/cgi-bin/imagemap/path/to/page.map`. Under Apache this has to read just `/path/to/page.map`.

Solution:

We use a global rule to remove the prefix on-the-fly for all requests:

```
RewriteEngine  on
RewriteRule    ^/cgi-bin/imagemap(.*)  $1  [PT]
```

Search pages in more than one directory

Description:

Sometimes it is neccessary to let the webserver search for pages in more than one directory. Here MultiViews or other techniques cannot help.

Solution:

We program a explicit ruleset which searches for the files in the directories.

```
RewriteEngine on

#   first try to find it in custom/...
#   ...and if found stop and be happy:
RewriteCond         /your/docroot/dir1/%{REQUEST_FILENAME}  -f
RewriteRule  ^(.+)  /your/docroot/dir1/$1  [L]

#   second try to find it in pub/...
#   ...and if found stop and be happy:
RewriteCond         /your/docroot/dir2/%{REQUEST_FILENAME}  -f
RewriteRule  ^(.+)  /your/docroot/dir2/$1  [L]

#   else go on for other Alias or ScriptAlias directives,
#   etc.
RewriteRule   ^(.+)  -  [PT]
```

Set Environment Variables According To URL Parts

Description:

Perhaps you want to keep status information between requests and use the URL to encode it. But you don't want to use a CGI wrapper for all pages just to strip out this

information.

Solution:

We use a rewrite rule to strip out the status information and remember it via an environment variable which can be later dereferenced from within XSSI or CGI. This way a URL `/foo/S=java/bar/` gets translated to `/foo/bar/` and the environment variable named `STATUS` is set to the value "java".

```
RewriteEngine on
RewriteRule   ^(.*)/S=([^/]+)/(.*)     $1/$3 [E=STATUS:$2]
```

Virtual User Hosts

Description:

Assume that you want to provide `www.username.host.domain.com` for the homepage of username via just DNS A records to the same machine and without any virtualhosts on this machine.

Solution:

For HTTP/1.0 requests there is no solution, but for HTTP/1.1 requests which contain a Host: HTTP header we can use the following ruleset to rewrite `http://www.username.host.com/anypath` internally to `/home/username/anypath`:

```
RewriteEngine on
RewriteCond   %{HTTP_HOST}                   ^www\.[^.]+\.host\.com$
RewriteRule   ^(.+)                          %{HTTP_HOST}$1          [C]
RewriteRule   ^www\.([^.]+)\.host\.com(.*) /home/$1$2
```

Redirect Homedirs For Foreigners

Description:

We want to redirect homedir URLs to another webserver `www.somewhere.com` when the requesting user does not stay in the local domain `ourdomain.com`. This is sometimes used in virtual host contexts.

Solution:

Just a rewrite condition:

```
RewriteEngine on
RewriteCond   %{REMOTE_HOST}  !^.+\.ourdomain\.com$
RewriteRule   ^(/~.+)         http://www.somewhere.com/$1 [R,L]
```

Redirect Failing URLs To Other Webserver

Description:

A typical FAQ about URL rewriting is how to redirect failing requests on webserver A to webserver B. Usually this is done via ErrorDocument CGI-scripts in Perl, but there is also a mod_rewrite solution. But notice that this is less performant than using a ErrorDocument CGI-script!

Solution:

The first solution has the best performance but less flexibility and is less error safe:

```
RewriteEngine on
RewriteCond   /your/docroot/%{REQUEST_FILENAME} !-f
```

```
RewriteRule   ^(.+)                              http://webserverB.dom/$1
```

The problem here is that this will only work for pages inside the DocumentRoot. While you can add more Conditions (for instance to also handle homedirs, etc.) there is better variant:

```
RewriteEngine on
RewriteCond   %{REQUEST_URI} !-U
RewriteRule   ^(.+)          http://webserverB.dom/$1
```

This uses the URL look-ahead feature of mod_rewrite. The result is that this will work for all types of URLs and is a safe way. But it does a performance impact on the webserver, because for every request there is one more internal subrequest. So, if your webserver runs on a powerful CPU, use this one. If it is a slow machine, use the first approach or better a ErrorDocument CGI-script.

Extended Redirection

Description:

Sometimes we need more control (concerning the character escaping mechanism) of URLs on redirects. Usually the Apache kernels URL escape function also escapes anchors, i.e. URLs like "url#anchor". You cannot use this directly on redirects with mod_rewrite because the uri_escape() function of Apache would also escape the hash character. How can we redirect to such a URL?

Solution:

We have to use a kludge by the use of a NPH-CGI script which does the redirect itself. Because here no escaping is done (NPH=non-parseable headers). First we introduce a new URL scheme `xredirect:` by the following per-server config-line (should be one of the last rewrite rules):

```
RewriteRule ^xredirect:(.+) /path/to/nph-xredirect.cgi/$1 \
            [T=application/x-httpd-cgi,L]
```

This forces all URLs prefixed with `xredirect:` to be piped through the `nph-xredirect.cgi` program. And this program just looks like:

```
#!/path/to/perl
##
##  nph-xredirect.cgi -- NPH/CGI script for extended redirects
##  Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved.
##

$| = 1;
$url = $ENV{'PATH_INFO'};

print "HTTP/1.0 302 Moved Temporarily\n";
print "Server: $ENV{'SERVER_SOFTWARE'}\n";
print "Location: $url\n";
print "Content-type: text/html\n";
print "\n";
print "<html>\n";
print "<head>\n";
```

```
print "<title>302 Moved Temporarily (EXTENDED)</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>Moved Temporarily (EXTENDED)</h1>\n";
print "The document has moved <a HREF=\"$url\">here</a>.<p>\n";
print "</body>\n";
print "</html>\n";

##EOF##
```

This provides you with the functionality to do redirects to all URL schemes, i.e. including the one which are not directly accepted by mod_rewrite. For instance you can now also redirect to `news:newsgroup` via

```
RewriteRule ^anyurl  xredirect:news:newsgroup
```

Notice: You have not to put [R] or [R,L] to the above rule because the `xredirect:` need to be expanded later by our special "pipe through" rule above.

Archive Access Multiplexer

Description:

Do you know the great CPAN (Comprehensive Perl Archive Network) under http://www.perl.com/CPAN? This does a redirect to one of several FTP servers around the world which carry a CPAN mirror and is approximately near the location of the requesting client. Actually this can be called an FTP access multiplexing service. While CPAN runs via CGI scripts, how can a similar approach implemented via mod_rewrite?

Solution:

First we notice that from version 3.0.0 mod_rewrite can also use the "ftp:" scheme on redirects. And second, the location approximation can be done by a rewritemap over the top-level domain of the client. With a tricky chained ruleset we can use this top-level domain as a key to our multiplexing map.

```
RewriteEngine on
RewriteMap    multiplex                 txt:/path/to/map.cxan
RewriteRule   ^/CxAN/(.*)               %{REMOTE_HOST}::$1
RewriteRule   ^.+\.([a-zA-Z]+)::(.*)$   ${multiplex:$1|ftp.default.dom}$2

##
##  map.cxan -- Multiplexing Map for CxAN
##

de          ftp://ftp.cxan.de/CxAN/
uk          ftp://ftp.cxan.uk/CxAN/
com         ftp://ftp.cxan.com/CxAN/
 :
##EOF##
```

Time-Dependend Rewriting

Description:

When tricks like time-dependend content should happen a lot of webmasters still use

CGI scripts which do for instance redirects to specialized pages. How can it be done via mod_rewrite?

Solution:

There are a lot of variables named `TIME_xxx` for rewrite conditions. In conjunction with the special lexicographic comparison patterns <STRING, >STRING and =STRING we can do time-dependend redirects:

```
RewriteEngine on
RewriteCond   %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond   %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule   ^foo\.html$             foo.day.html
RewriteRule   ^foo\.html$             foo.night.html
```

This provides the content of `foo.day.html` under the URL `foo.html` from 07:00-19:00 and at the remaining time the contents of `foo.night.html`. Just a nice feature for a homepage...

Backward Compatibility for YYYY to XXXX migration

Description:

How can we make URLs backward compatible (still existing virtually) after migrating document.YYYY to document.XXXX, e.g. after translating a bunch of .html files to .phtml?

Solution:

We just rewrite the name to its basename and test for existence of the new extension. If it exists, we take that name, else we rewrite the URL to its original state.

```
#   backward compatibility ruleset for
#   rewriting document.html to document.phtml
#   when and only when document.phtml exists
#   but no longer document.html
RewriteEngine on
RewriteBase   /~quux/
#   parse out basename, but remember the fact
RewriteRule   ^(.*)\.html$              $1      [C,E=WasHTML:yes]
#   rewrite to document.phtml if exists
RewriteCond   %{REQUEST_FILENAME}.phtml -f
RewriteRule   ^(.*)$ $1.phtml                   [S=1]
#   else reverse the previous basename cutout
RewriteCond   %{ENV:WasHTML}            ^yes$
RewriteRule   ^(.*)$ $1.html
```

Content Handling

From Old to New (intern)

Description:

Assume we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. Actually we want that users of the old URL even not recognize that the pages was renamed.

Solution:

We rewrite the old URL to the new one internally via the following rule:

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo\.html$  bar.html
```

From Old to New (extern)

Description:

Assume again that we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.

Solution:

We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo\.html$  bar.html  [R]
```

Browser Dependend Content

Description:

At least for important top-level pages it is sometimes necesarry to provide the optimum of browser dependend content, i.e. one has to provide a maximum version for the latest Netscape variants, a minimum version for the Lynx browsers and a average feature version for all others.

Solution:

We cannot use content negotiation because the browsers do not provide their type in that form. Instead we have to act on the HTTP header "User-Agent". The following condig does the following: If the HTTP header "User-Agent" begins with "Mozilla/3", the page `foo.html` is rewritten to `foo.NS.html` and and the rewriting stops. If the browser is "Lynx" or "Mozilla" of version 1 or 2 the URL becomes `foo.20.html`. All other browsers receive page `foo.32.html`. This is done by the following ruleset:

```
RewriteCond %{HTTP_USER_AGENT}  ^Mozilla/3.*
RewriteRule ^foo\.html$         foo.NS.html          [L]

RewriteCond %{HTTP_USER_AGENT}  ^Lynx/.*          [OR]
RewriteCond %{HTTP_USER_AGENT}  ^Mozilla/[12].*
RewriteRule ^foo\.html$         foo.20.html          [L]

RewriteRule ^foo\.html$         foo.32.html          [L]
```

Dynamic Mirror

Description:

Assume there are nice webpages on remote hosts we want to bring into our namespace. For FTP servers we would use the `mirror` program which actually maintains an explicit up-to-date copy of the remote data on the local machine. For a webserver we could use the program `webcopy` which acts similar via HTTP. But both

techniques have one major drawback: The local copy is always just as up-to-date as often we run the program. It would be much better if the mirror is not a static one we have to establish explicitly. Instead we want a dynamic mirror with data which gets updated automatically when there is need (updated data on the remote host).

Solution:

To provide this feature we map the remote webpage or even the complete remote webarea to our namespace by the use of the *Proxy Throughput* feature (flag [P]):

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^hotsheet/(.*)$  http://www.tstimpreso.com/hotsheet/$1  [P

RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^usa-news\.html$  http://www.quux-corp.com/news/index.htm
```

Reverse Dynamic Mirror

Description:

...

Solution:

```
RewriteEngine on
RewriteCond   /mirror/of/remotesite/$1           -U
RewriteRule   ^http://www\.remotesite\.com/(.*)$ /mirror/of/remotesite/$1
```

Retrieve Missing Data from Intranet

Description:

This is a tricky way of virtually running a corporates (external) Internet webserver (`www.quux-corp.dom`), while actually keeping and maintaining its data on a (internal) Intranet webserver (`www2.quux-corp.dom`) which is protected by a firewall. The trick is that on the external webserver we retrieve the requested data on-the-fly from the internal one.

Solution:

First, we have to make sure that our firewall still protects the internal webserver and that only the external webserver is allowed to retrieve data from it. For a packet-filtering firewall we could for instance configure a firewall ruleset like the following:

```
ALLOW Host www.quux-corp.dom Port >1024 --> Host www2.quux-corp.dom Port
DENY  Host *                 Port *     --> Host www2.quux-corp.dom Port
```

Just adjust it to your actual configuration syntax. Now we can establish the mod_rewrite rules which request the missing data in the background through the proxy throughput feature:

```
RewriteRule ^/~([^/]+)/?(.*)          /home/$1/.www/$2
RewriteCond %{REQUEST_FILENAME}       !-f
RewriteCond %{REQUEST_FILENAME}       !-d
RewriteRule ^/home/([^/]+)/.www/?(.*) http://www2.quux-corp.dom/~$1/pub/$
```

Load Balancing

Description:

Suppose we want to load balance the traffic to `www.foo.com` over `www[0-5].foo.com` (a total of 6 servers). How can this be done?

Solution:

There are a lot of possible solutions for this problem. We will discuss first a commonly known DNS-based variant and then the special one with mod_rewrite:

1. **DNS Round-Robin**

   The simplest method for load-balancing is to use the DNS round-robin feature of BIND. Here you just configure `www[0-9].foo.com` as usual in your DNS with A(address) records, e.g.

```
www0    IN  A       1.2.3.1
www1    IN  A       1.2.3.2
www2    IN  A       1.2.3.3
www3    IN  A       1.2.3.4
www4    IN  A       1.2.3.5
www5    IN  A       1.2.3.6
```

   Then you additionally add the following entry:

```
www     IN  CNAME   www0.foo.com.
        IN  CNAME   www1.foo.com.
        IN  CNAME   www2.foo.com.
        IN  CNAME   www3.foo.com.
        IN  CNAME   www4.foo.com.
        IN  CNAME   www5.foo.com.
        IN  CNAME   www6.foo.com.
```

   Notice that this seems wrong, but is actually an intended feature of BIND and can be used in this way. However, now when `www.foo.com` gets resolved, BIND gives out `www0-www6` - but in a slightly permutated/rotated order every time. This way the clients are spread over the various servers. But notice that this not a perfect load balancing scheme, because DNS resolve information gets cached by the other nameservers on the net, so once a client has resolved `www.foo.com` to a particular `wwwN.foo.com`, all subsequent requests also go to this particular name `wwwN.foo.com`. But the final result is ok, because the total sum of the requests are really spread over the various webservers.

2. **DNS Load-Balancing**

   A sophisticated DNS-based method for load-balancing is to use the program `lbnamed` which can be found at http://www.stanford.edu/~schemers/docs/lbnamed/lbnamed.html. It is a Perl 5 program in conjunction with auxilliary tools which provides a real load-balancing for DNS.

3. **Proxy Throughput Round-Robin**

   In this variant we use mod_rewrite and its proxy throughput feature. First we dedicate `www0.foo.com` to be actually `www.foo.com` by using a single

```
www     IN  CNAME    www0.foo.com.
```

entry in the DNS. Then we convert `www0.foo.com` to a proxy-only server, i.e. we configure this machine so all arriving URLs are just pushed through the internal proxy to one of the 5 other servers (`www1-www5`). To accomplish this we first establish a ruleset which contacts a load balancing script `lb.pl` for all URLs.

```
RewriteEngine on
RewriteMap    lb      prg:/path/to/lb.pl
RewriteRule   ^/(.+)$ ${lb:$1}           [P,L]
```

Then we write `lb.pl`:

```
#!/path/to/perl
##
##  lb.pl -- load balancing script
##

$| = 1;

$name   = "www";     # the hostname base
$first  = 1;         # the first server (not 0 here, because 0 is my
$last   = 5;         # the last server in the round-robin
$domain = "foo.dom"; # the domainname

$cnt = 0;
while (<STDIN>) {
    $cnt = (($cnt+1) % ($last+1-$first));
    $server = sprintf("%s%d.%s", $name, $cnt+$first, $domain);
    print "http://$server/$_";
}

##EOF##
```

A last notice: Why is this useful? Seems like `www0.foo.com` still is overloaded? The answer is yes, it is overloaded, but with plain proxy throughput requests, only! All SSI, CGI, ePerl, etc. processing is completely done on the other machines. This is the essential point.

4. **Hardware/TCP Round-Robin**

   There is a hardware solution available, too. Cisco has a beast called LocalDirector which does a load balancing at the TCP/IP level. Actually this is some sort of a circuit level gateway in front of a webcluster. If you have enough money and really need a solution with high performance, use this one.

New MIME-type, New Service

Description:

On the net there are a lot of nifty CGI programs. But their usage is usually boring, so a lot of webmaster don't use them. Even Apache's Action handler feature for MIME-

types is only appropriate when the CGI programs don't need special URLs (actually PATH_INFO and QUERY_STRINGS) as their input. First, let us configure a new file type with extension `.scgi` (for secure CGI) which will be processed by the popular `cgiwrap` program. The problem here is that for instance we use a Homogeneous URL Layout (see above) a file inside the user homedirs has the URL `/u/user/foo/bar.scgi`. But `cgiwrap` needs the URL in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uge]/([^/]+)/\.www/(.+)\.scgi(.*) ...
... /internal/cgi/user/cgiwrap/~$1/$2.scgi$3  [NS,T=application/x-http-cg
```

Or assume we have some more nifty programs: `wwwlog` (which displays the `access.log` for a URL subtree and `wwwidx` (which runs Glimpse on a URL subtree). We have to provide the URL area to these programs so they know on which area they have to act on. But usually this ugly, because they are all the times still requested from that areas, i.e. typically we would run the `swwidx` program from within `/u/user/foo/` via hyperlink to

```
/internal/cgi/user/swwidx?i=/u/user/foo/
```

which is ugly. Because we have to hard-code **both** the location of the area **and** the location of the CGI inside the hyperlink. When we have to reorganise or area, we spend a lot of time changing the various hyperlinks.

Solution:

The solution here is to provide a special new URL format which automatically leads to the proper CGI invocation. We configure the following:

```
RewriteRule   ^/([uge])/([^/]+)(/?.*)/\*   /internal/cgi/user/wwwidx?i=/$1,
RewriteRule   ^/([uge])/([^/]+)(/?.*):log  /internal/cgi/user/wwwlog?f=/$1,
```

Now the hyperlink to search at `/u/user/foo/` reads only

```
HREF="*"
```

which internally gets automatically transformed to

```
/internal/cgi/user/wwwidx?i=/u/user/foo/
```

The same approach leads to an invocation for the access log CGI program when the hyperlink `:log` gets used.

From Static to Dynamic

Description:

How can we transform a static page `foo.html` into a dynamic variant `foo.cgi` in a seamless way, i.e. without notice by the browser/user.

Solution:

We just rewrite the URL to the CGI-script and force the correct MIME-type so it gets really run as a CGI-script. This way a request to `/~quux/foo.html` internally leads to the invokation of `/~quux/foo.cgi`.

```
RewriteEngine  on
RewriteBase    /~quux/
RewriteRule    ^foo\.html$  foo.cgi  [T=application/x-httpd-cgi]
```

On-the-fly Content-Regeneration

Description:

Here comes a really esoteric feature: Dynamically generated but statically served pages, i.e. pages should be delivered as pure static pages (read from the filesystem and just passed through), but they have to be generated dynamically by the webserver if missing. This way you can have CGI-generated pages which are statically served unless one (or a cronjob) removes the static contents. Then the contents gets refreshed.

Solution:

This is done via the following ruleset:

```
RewriteCond %{REQUEST_FILENAME}   !-s
RewriteRule ^page\.html$          page.cgi   [T=application/x-httpd-cgi,L]
```

Here a request to `page.html` leads to a internal run of a corresponding `page.cgi` if `page.html` is still missing or has filesize null. The trick here is that `page.cgi` is a usual CGI script which (additionally to its STDOUT) writes its output to the file `page.html`. Once it was run, the server sends out the data of `page.html`. When the webmaster wants to force a refresh the contents, he just removes `page.html` (usually done by a cronjob).

Document With Autorefresh

Description:

Wouldn't it be nice while creating a complex webpage if the webbrowser would automatically refresh the page every time we write a new version from within our editor? Impossible?

Solution:

No! We just combine the MIME multipart feature, the webserver NPH feature and the URL manipulation power of mod_rewrite. First, we establish a new URL feature: Adding just `:refresh` to any URL causes this to be refreshed every time it gets updated on the filesystem.

```
RewriteRule   ^(/[uge]/[^/]+/?.*):refresh  /internal/cgi/apache/nph-refres
```

Now when we reference the URL

```
/u/foo/bar/page.html:refresh
```

this leads to the internal invocation of the URL

```
/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html
```

The only missing part is the NPH-CGI script. Although one would usually say "left as an exercise to the reader" ;-) I will provide this, too.

```
#!/sw/bin/perl
##
##  nph-refresh -- NPH/CGI script for auto refreshing pages
##  Copyright (c) 1997 Ralf S. Engelschall, All Rights Reserved.
##
$| = 1;

#   split the QUERY_STRING variable
@pairs = split(/&/, $ENV{'QUERY_STRING'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
    eval "\$$name = \"$value\"";
}
$QS_s = 1 if ($QS_s eq '');
$QS_n = 3600 if ($QS_n eq '');
if ($QS_f eq '') {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n";
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS_f not found\n";
    exit(0);
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace;boundary=$bound\n";
    &print_http_headers_multipart_next;
}

sub print_http_headers_multipart_next {
    print "\n--$bound\n";
}

sub print_http_headers_multipart_end {
    print "\n--$bound--\n";
}

sub displayhtml {
    local($buffer) = @_;
    $len = length($buffer);
    print "Content-type: text/html\n";
    print "Content-length: $len\n\n";
    print $buffer;
}

sub readfile {
    local($file) = @_;
    local(*FP, $size, $buffer, $bytes);
    ($x, $x, $x, $x, $x, $x, $x, $size) = stat($file);
    $size = sprintf("%d", $size);
    open(FP, "&lt;$file");
```

```
    $bytes = sysread(FP, $buffer, $size);
    close(FP);
    return $buffer;
}

$buffer = &readfile($QS_f);
&print_http_headers_multipart_begin;
&displayhtml($buffer);

sub mystat {
    local($file) = $_[0];
    local($time);

    ($x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime) = stat($file);
    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n &lt; $QS_n; $n++) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ($mtime ne $mtimeL) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
            &displayhtml($buffer);
            sleep(5);
            $mtimeL = &mystat($QS_f);
            last;
        }
        sleep($QS_s);
    }
}

&print_http_headers_multipart_end;

exit(0);

##EOF##
```

Mass Virtual Hosting

Description:

The <VirtualHost> feature of Apache is nice and works great when you just have a few dozens virtual hosts. But when you are an ISP and have hundreds of virtual hosts to provide this feature is not the best choice.

Solution:

To provide this feature we map the remote webpage or even the complete remote webarea to our namespace by the use of the *Proxy Throughput* feature (flag [P]):

```
##
##  vhost.map
##
www.vhost1.dom:80  /path/to/docroot/vhost1
www.vhost2.dom:80  /path/to/docroot/vhost2
     :
```

```
www.vhostN.dom:80  /path/to/docroot/vhostN

##
##  httpd.conf
##
    :
#   use the canonical hostname on redirects, etc.
UseCanonicalName on

    :
#   add the virtual host in front of the CLF-format
CustomLog  /path/to/access_log  "%{VHOST}e %h %l %u %t \"%r\" %>s %b"
    :

#   enable the rewriting engine in the main server
RewriteEngine on

#   define two maps: one for fixing the URL and one which defines
#   the available virtual hosts with their corresponding
#   DocumentRoot.
RewriteMap    lowercase    int:tolower
RewriteMap    vhost        txt:/path/to/vhost.map

#   Now do the actual virtual host mapping
#   via a huge and complicated single rule:
#
#   1. make sure we don't map for common locations
RewriteCond   %{REQUEST_URI}  !^/commonurl1/.*
RewriteCond   %{REQUEST_URI}  !^/commonurl2/.*
    :
RewriteCond   %{REQUEST_URI}  !^/commonurlN/.*
#
#   2. make sure we have a Host header, because
#      currently our approach only supports
#      virtual hosting through this header
RewriteCond   %{HTTP_HOST}  !^$
#
#   3. lowercase the hostname
RewriteCond   ${lowercase:%{HTTP_HOST}|NONE}  ^(.+)$
#
#   4. lookup this hostname in vhost.map and
#      remember it only when it is a path
#      (and not "NONE" from above)
RewriteCond   ${vhost:%1}  ^(/.*)$
#
#   5. finally we can map the URL to its docroot location
#      and remember the virtual host for logging puposes
RewriteRule   ^/(.*)$   %1/$1  [E=VHOST:${lowercase:%{HTTP_HOST}}]
    :
```

Access Restriction

Blocking of Robots

Description:

How can we block a really annoying robot from retrieving pages of a specific

webarea? A `/robots.txt` file containing entries of the "Robot Exclusion Protocol" is typically not enough to get rid of such a robot.

Solution:

We use a ruleset which forbids the URLs of the webarea `/~quux/foo/arc/` (perhaps a very deep directory indexed area where the robot traversal would create big server load). We have to make sure that we forbid access only to the particular robot, i.e. just forbidding the host where the robot runs is not enough. This would block users from this host, too. We accomplish this by also matching the User-Agent HTTP header information.

```
RewriteCond %{HTTP_USER_AGENT}   ^NameOfBadRobot.*
RewriteCond %{REMOTE_ADDR}       ^123\.45\.67\.[8-9]$
RewriteRule ^/~quux/foo/arc/.+   -   [F]
```

Blocked Inline-Images

Description:

Assume we have under http://www.quux-corp.de/~quux/ some pages with inlined GIF graphics. These graphics are nice, so others directly incorporate them via hyperlinks to their pages. We don't like this practice because it adds useless traffic to our server.

Solution:

While we cannot 100% protect the images from inclusion, we can at least restrict the cases where the browser sends a HTTP Referer header.

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://www.quux-corp.de/~quux/.*$ [NC]
RewriteRule .*\.gif$        -                                   [F]

RewriteCond %{HTTP_REFERER}         !^$
RewriteCond %{HTTP_REFERER}         !.*/foo-with-gif\.html$
RewriteRule ^inlined-in-foo\.gif$   -                       [F]
```

Host Deny

Description:

How can we forbid a list of externally configured hosts from using our server?

Solution:

For Apache >= 1.3b6:

```
RewriteEngine on
RewriteMap    hosts-deny  txt:/path/to/hosts.deny
RewriteCond   ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND} !=NOT-FOUND [OR]
RewriteCond   ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND} !=NOT-FOUND
RewriteRule   ^/.*  -  [F]
```

For Apache <= 1.3b6:

```
RewriteEngine on
RewriteMap    hosts-deny  txt:/path/to/hosts.deny
RewriteRule   ^/(.*)$ ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND}/$1
RewriteRule   !^NOT-FOUND/.* - [F]
RewriteRule   ^NOT-FOUND/(.*)$ ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND}/$1
RewriteRule   !^NOT-FOUND/.* - [F]
```

```
RewriteRule   ^NOT-FOUND/(.*)$ /$1

##
##  hosts.deny
##
##  ATTENTION! This is a map, not a list, even when we treat it as such.
##             mod_rewrite parses it for key/value pairs, so at least a
##             dummy value "-" must be present for each entry.
##

193.102.180.41 -
bsdti1.sdm.de  -
192.76.162.40  -
```

URL-Restricted Proxy

Description:

How can we restrict the proxy to allow access to a configurable set of internet sites only? The site list is extracted from a prepared bookmarks file.

Solution:

We first have to make sure mod_rewrite is below(!) mod_proxy in the `Configuration` file when compiling the Apache webserver (or in the `AddModule` list of `httpd.conf` in the case of dynamically loaded modules), as it must get called *_before_* mod_proxy.

For simplicity, we generate the site list as a textfile map (but see the mod_rewrite documentation for a conversion script to DBM format). A typical Netscape bookmarks file can be converted to a list of sites with a shell script like this:

```
#!/bin/sh
cat ${1:-~/.netscape/bookmarks.html} |
tr -d '\015' | tr '[A-Z]' '[a-z]' | grep href=\" |
sed -e '/href="file:/d;' -e '/href="news:/d;' \
    -e 's|^.*href="[^:]*://\([^:/"]*\).*$|\1 OK|;' \
    -e '/href="/s|^.*href="\([^:/"]*\).*$|\1 OK|;' |
sort -u
```

We redirect the resulting output into a text file called `goodsites.txt`. It now looks similar to this:

```
www.apache.org OK
xml.apache.org OK
jakarta.apache.org OK
perl.apache.org OK
...
```

We reference this site file within the configuration for the `VirtualHost` which is responsible for serving as a proxy (often not port 80, but 81, 8080 or 8008).

```
<VirtualHost *:8008>
  ...
  RewriteEngine   On
```

```
  # Either use the (plaintext) allow list from goodsites.txt
  RewriteMap      ProxyAllow   txt:/usr/local/apache/conf/goodsites.txt
  # Or, for faster access, convert it to a DBM database:
  #RewriteMap     ProxyAllow   dbm:/usr/local/apache/conf/goodsites
  # Match lowercased hostnames
  RewriteMap      lowercase    int:tolower
  # Here we go:
  # 1) first lowercase the site name and strip off a :port suffix
  RewriteCond  ${lowercase:%{HTTP_HOST}}    ^([^:]*).*$
  # 2) next look it up in the map file.
  #    "%1" refers to the previous regex.
  #    If the result is "OK", proxy access is granted.
  RewriteCond  ${ProxyAllow:%1|DENY}        !^OK$          [NC]
  # 3) Disallow proxy requests if the site was _not_ tagged "OK":
  RewriteRule  ^proxy:                      -              [F]
  ...
</VirtualHost>
```

Proxy Deny

Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

Solution:

We first have to make sure mod_rewrite is below(!) mod_proxy in the `Configuration` file when compiling the Apache webserver. This way it gets called *_before_* mod_proxy. Then we configure the following for a host-dependend deny...

```
RewriteCond %{REMOTE_HOST} ^badhost\.mydomain\.com$
RewriteRule !^http://[^/.]\.mydomain.com.*  - [F]
```

...and this one for a user@host-dependend deny:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST}  ^badguy@badhost\.mydomain\.co
RewriteRule !^http://[^/.]\.mydomain.com.*  - [F]
```

Special Authentication Variant

Description:

Sometimes a very special authentication is needed, for instance a authentication which checks for a set of explicitly configured users. Only these should receive access and without explicit prompting (which would occur when using the Basic Auth via mod_access).

Solution:

We use a list of rewrite conditions to exclude all except our friends:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend1@client1.quux-corp\.c
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend2@client2.quux-corp\.c
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend3@client3.quux-corp\.c
RewriteRule ^/~quux/only-for-friends/      -
```

Referer-based Deflector

Description:

How can we program a flexible URL Deflector which acts on the "Referer" HTTP header and can be configured with as many referring pages as we like?

Solution:

Use the following really tricky ruleset...

```
RewriteMap  deflector txt:/path/to/deflector.map

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}} ^-$
RewriteRule ^.* %{HTTP_REFERER} [R,L]

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND} !=NOT-FOUND
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... in conjunction with a corresponding rewrite map:

```
##
##  deflector.map
##

http://www.badguys.com/bad/index.html     -
http://www.badguys.com/bad/index2.html    -
http://www.badguys.com/bad/index3.html    http://somewhere.com/
```

This automatically redirects the request back to the referring page (when "-" is used as the value in the map) or to a specific URL (when an URL is specified in the map as the second argument).

Other

External Rewriting Engine

Description:

A FAQ: How can we solve the FOO/BAR/QUUX/etc. problem? There seems no solution by the use of mod_rewrite...

Solution:

Use an external rewrite map, i.e. a program which acts like a rewrite map. It is run once on startup of Apache receives the requested URLs on STDIN and has to put the resulting (usually rewritten) URL on STDOUT (same order!).

```
RewriteEngine on
RewriteMap    quux-map       prg:/path/to/map.quux.pl
RewriteRule   ^/~quux/(.*)$  /~quux/${quux-map:$1}

#!/path/to/perl

#   disable buffered I/O which would lead
#   to deadloops for the Apache server
$| = 1;

#   read URLs one per line from stdin and
#   generate substitution URL on stdout
```

```
while (<>) {
    s|^foo/|bar/|;
    print $_;
}
```

This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such maps can be **used** also by an average user, only the system administrator can **define** it.

Apache HTTP Server Version 1.3